16. A Formal Modelling Language Extending SysML for Simulation of Continuous and Discrete System

Mark Hodson[1] and Nick Luckman[2]
[1]Block Software and [2]Weapons Systems Division, DSTO

Abstract

MBSE tools and techniques in a broad sense provide a structured approach to developing conceptual models of complex systems. Key features of these approaches are: the use of graphical based views on a central model that reflect the interests of particular stakeholders in the system; hierarchical decomposition of the system in question; and an ability to add, over time, increasing levels of detail to the model as knowledge is acquired, or in other words allow the model to move from the abstract towards the formal without the need to redefine the model in a different modelling environment. Through such an approach the leap of faith required to transition from model to real system is reduced when compared to traditional techniques.

When the real world system is software it is possible to take the conceptual modelling methodologies all the way to a formal (in the mathematical sense) specification such that ultimately the model has a one to one mapping with the real software system. Indeed great strides have been made with modelling methodologies and tools in the software domain, for example with UML.

Systems Engineering of course has to deal with complex application domains well beyond just software, where any model of the system will always be conceptual at some level because a one to one mapping with the real system will never exist. SysML is an extension and modification of UML that aims to support the broader modelling needs of SE, hence the term MBSE. However, engineering has at its disposal another type of modelling that is simulation, which can provide great insights into the behaviour of complex systems. Although UML and SysML primarily support conceptual modelling they do have enough formality in them to support certain types of simulation (after all computer based simulations are in themselves software systems), for example in some behavioural graphical views, such as activity and state machine diagrams. The algorithmic model of computation used with these is basically Discrete Event Simulation (DEVS) such that the transitions between activities or state represent discrete events in time. Although many systems can adequately be simulated with discrete events (in time) many more need more powerful models of computation such as discrete time and Ordinary Differential Equation (ODE) solving, which although can be expressed in the DEVS formalisms are generally only realised in specialised engineering level, graphical based, modelling and simulation tools such as Simulink®. Such tools are built principally first and foremost to create formal models in a bottom up approach and thus lack features to support for conceptual modelling.

Interestingly the diagrams used in specialised engineering M&S tools often have the appearance of structural models. This is because they are actually graphical representations of mathematical algorithms, more precisely iterative algorithms. The challenge therefore for MBSE is to develop general purpose graphical modelling views that transition naturally from

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
FEB 2013	**N/A**	**-**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
A Formal Modelling Language Extending SysML for Simulation of Continuous and Discrete System	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Block Software and Weapons Systems Division, DSTO**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited

13. SUPPLEMENTARY NOTES
See also ADA585222. Proceedings of the 2012 Model-Based Systems Engineering Symposium, 27 - 28 November 2012, DSTO Edinburgh, South Australia., The original document contains color images.

14. ABSTRACT
MBSE tools and techniques in a broad sense provide a structured approach to developing conceptual models of complex systems. Key features of these approaches are: the use of graphical based views on a central model that reflect the interests of particular stakeholders in the system; hierarchical decomposition of the system in question; and an ability to add, over time, increasing levels of detail to the model as knowledge is acquired, or in other words allow the model to move from the abstract towards the formal without the need to redefine the model in a different modelling environment. Through such an approach the leap of faith required to transition from model to real system is reduced when compared to traditional techniques.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**SAR**	**19**	

system relevant decomposition views into views of iterative algorithms capable of being executed with potentially any iterative model of computation.

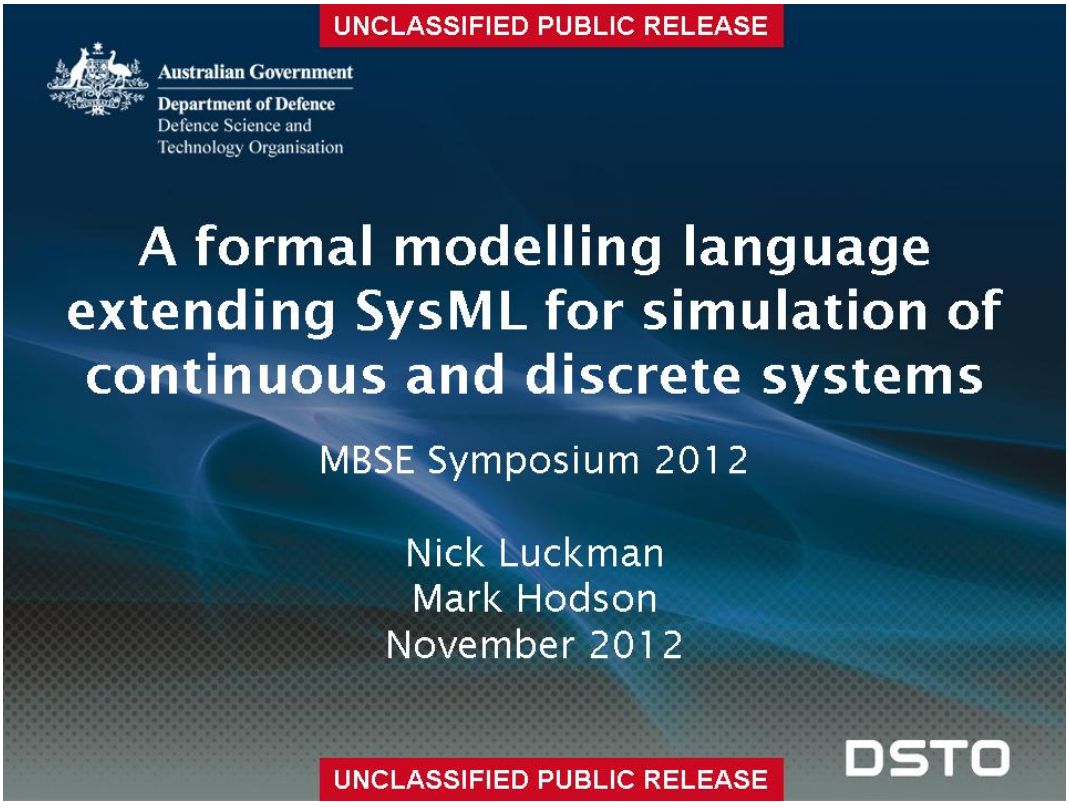
This paper outlines a graphical modelling view similar to the internal block diagram of SysML that supports hierarchical decomposition and iterative algorithmic expression at the same time.

Presenter Biography

Mark Hodson graduated with 1st class honours in Computer Systems Engineering from Adelaide University at the end of 1999. Since that time, Mark has worked for Tenix Electronic Systems Division (formerly Vision Abell, now BAE Systems) in the areas of information security and hydrography, and has spent much of the last 10 years working on contract in Weapons Systems Division in DSTO in the areas of M&S theory and accompanying architecture development, collaborative vulnerability and lethality models, and providing software engineering support to specific tasks within the branch.

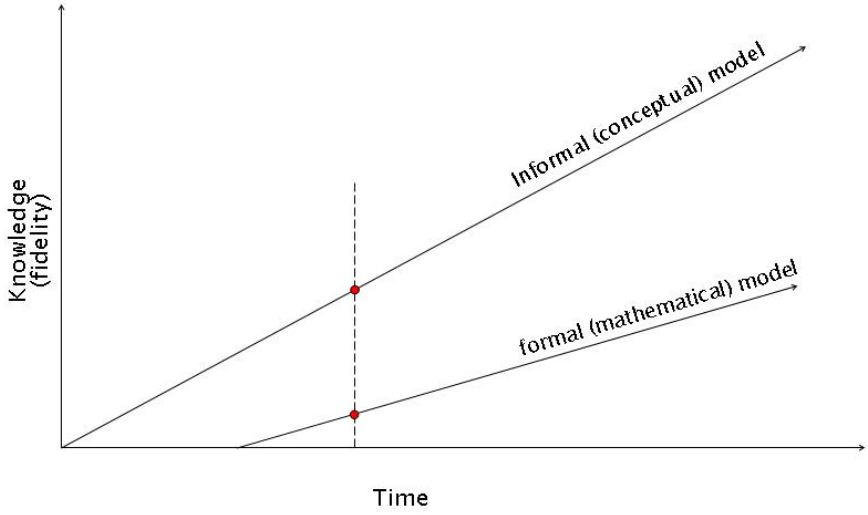
Nick Luckman graduated from Adelaide University in 1990 with a degree in Mechanical Engineering. Since then he has worked for the Defence Science and Technology Organisation working mostly on weapons systems. During this time he has developed many simulations with various levels of complexity and purpose. In the last seven years or so he has worked on developing modelling and simulation frameworks and architectures that take into account the business case of reuse.

Presentation



UNCLASSIFIED PUBLIC RELEASE

Background: Modelling builds knowledge

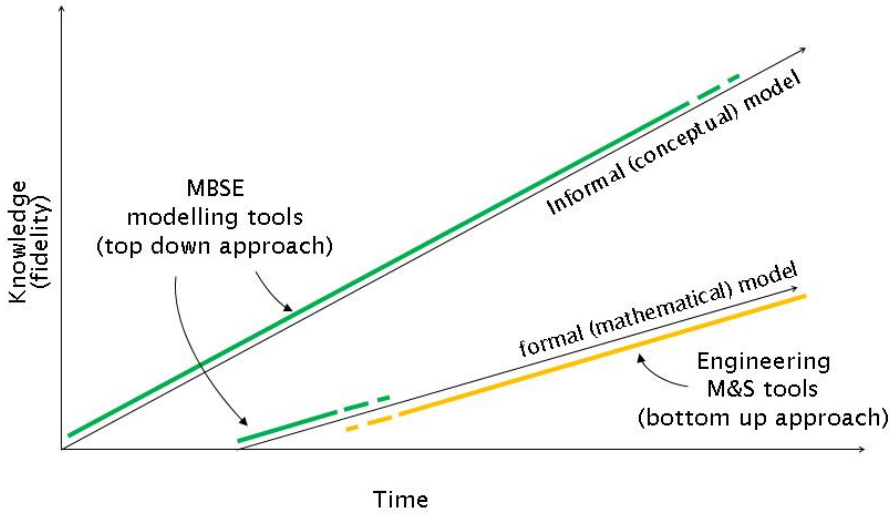


UNCLASSIFIED PUBLIC RELEASE

DSTO

UNCLASSIFIED PUBLIC RELEASE

Background: Different modelling approaches

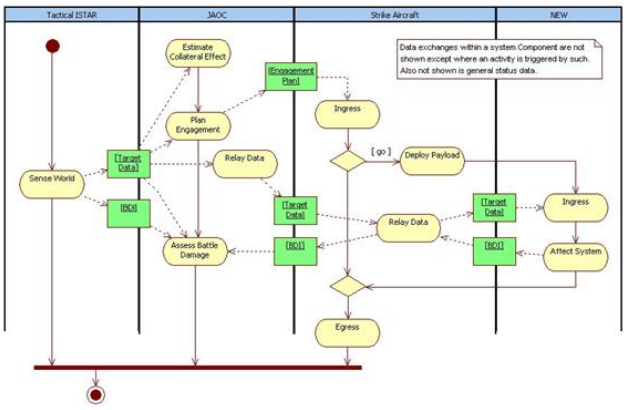


UNCLASSIFIED PUBLIC RELEASE

DSTO

UNCLASSIFIED PUBLIC RELEASE

MBSE Approach to modelling behaviour



- Questions are around when the events occur.
- What about questions of a non-temporal nature?
- Either way it may be necessary to simulate the continuous-time behaviour of the system.

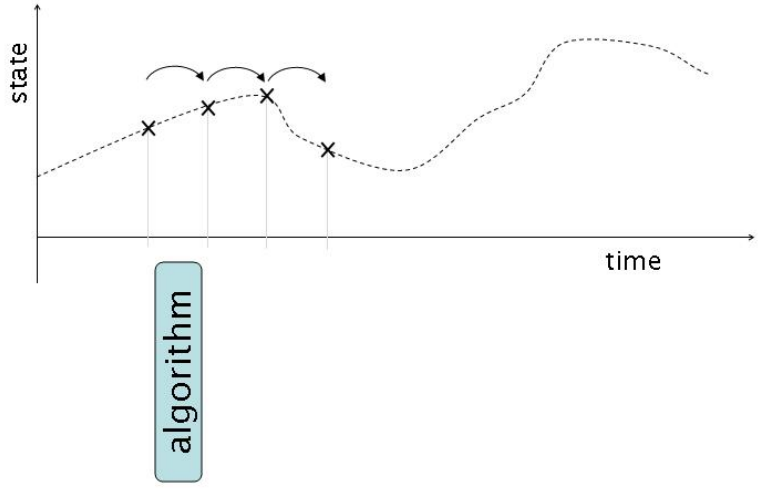
- The main elements represent constant behaviour for a period of time.
- Connections represent instantaneous transitions between different constant behaviours.
- This lends itself to simulating sequences of discrete events in time.

UNCLASSIFIED PUBLIC RELEASE

DSTO

UNCLASSIFIED PUBLIC RELEASE

Traditional approach to modelling behaviour

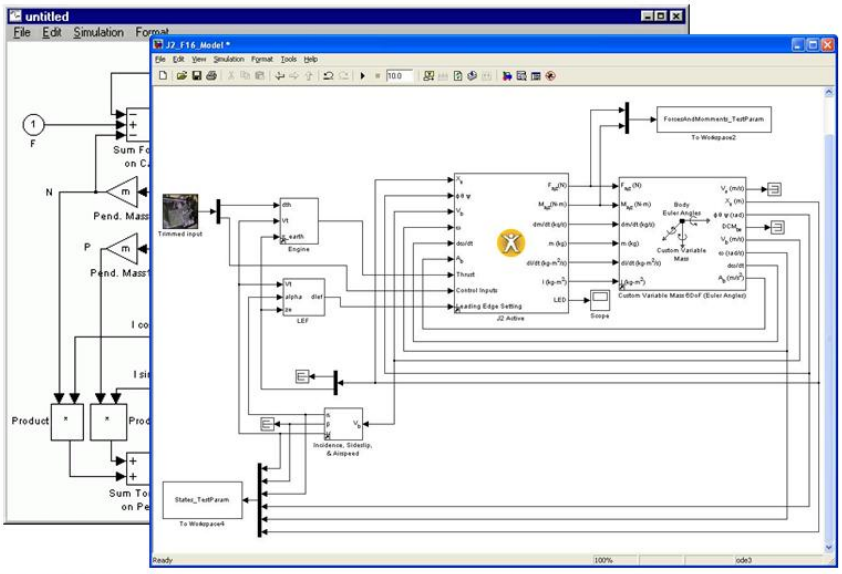


UNCLASSIFIED PUBLIC RELEASE

DSTO

UNCLASSIFIED PUBLIC RELEASE

Traditional algorithmic model

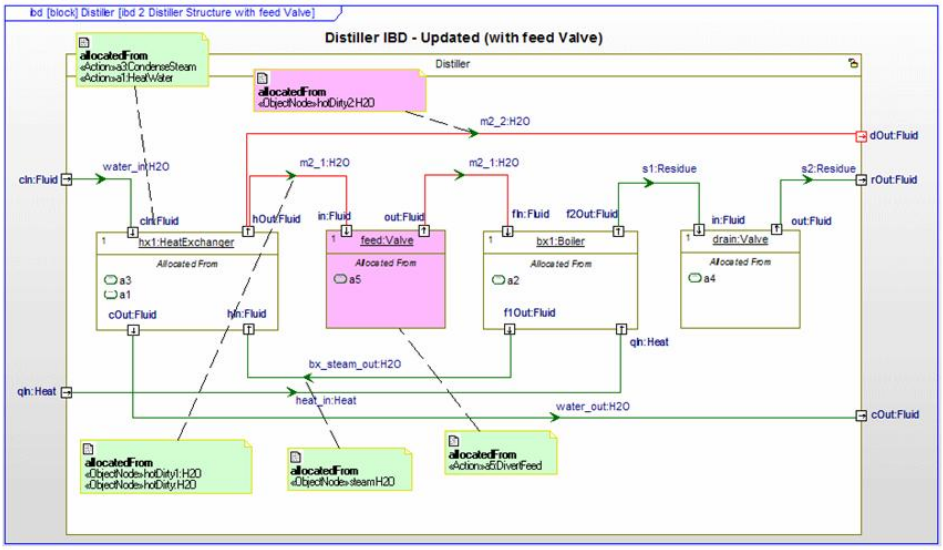


UNCLASSIFIED PUBLIC RELEASE

DSTO

UNCLASSIFIED PUBLIC RELEASE

MBSE Approach to modelling algorithm structure

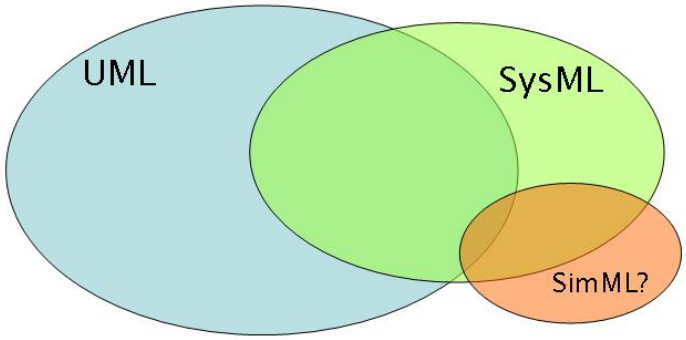


UNCLASSIFIED PUBLIC RELEASE

DSTO

UNCLASSIFIED PUBLIC RELEASE

Proposed Solution



UNCLASSIFIED PUBLIC RELEASE

DSTO

UNCLASSIFIED PUBLIC RELEASE

Summary of Goals

- Seamless transition from systems decomposition to algorithm decomposition.
  - Both are structural views.
- Means of integrating multiple Models of Computation (MoC):
  - Discrete Event;
  - Discrete Time;
  - ODE solving;
  - etc
- Means of encapsulating MoC within branches of a model decomposition (MoC within MoC).

UNCLASSIFIED PUBLIC RELEASE

DSTO

UNCLASSIFIED

Australian Government
Department of Defence
Defence Science and Technology Organisation

Underpinning M&S Theory

DSTO

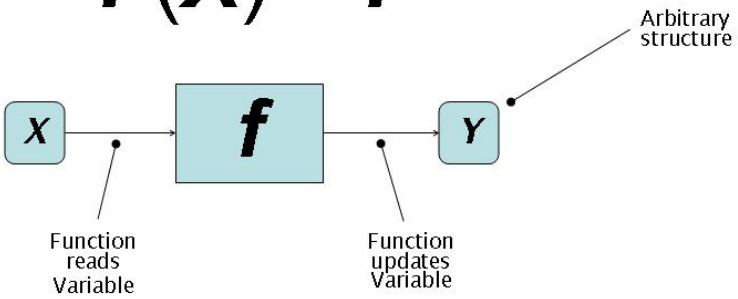
UNCLASSIFIED

UNCLASSIFIED PUBLIC RELEASE

Mathematical Algorithms for Simulation

- Algorithms are formulated in terms of Variables and mathematic operations upon them (Functions).

$$f(X) \rightarrow Y$$



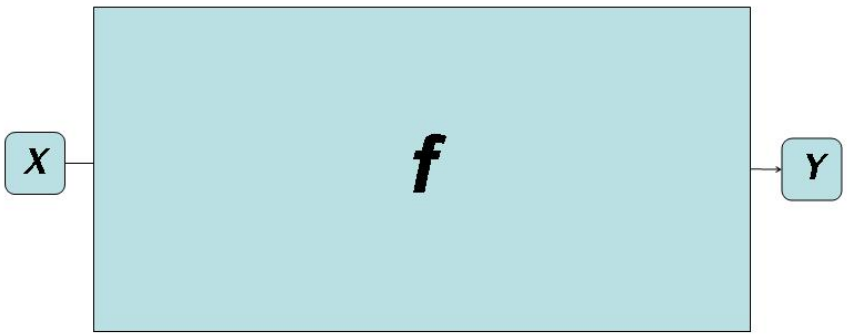
DSTO

UNCLASSIFIED PUBLIC RELEASE

UNCLASSIFIED PUBLIC RELEASE

Mathematical Algorithms for Simulation

- Functions can be decomposed.



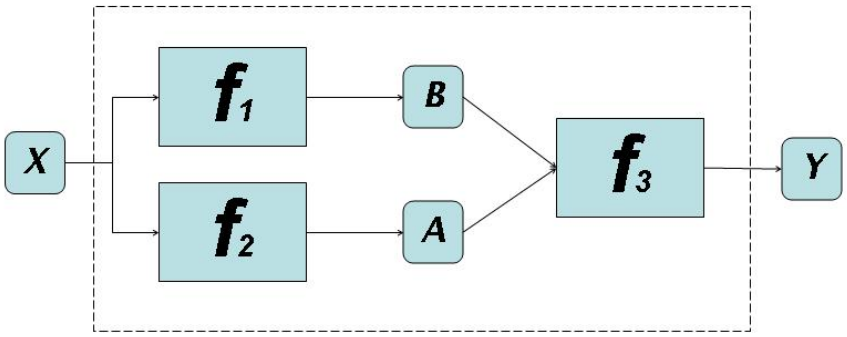
UNCLASSIFIED PUBLIC RELEASE

DSTO

UNCLASSIFIED PUBLIC RELEASE

Mathematical Algorithms for Simulation

- Functions can be decomposed.



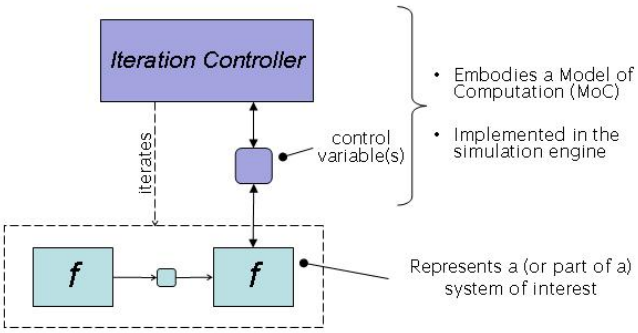
UNCLASSIFIED PUBLIC RELEASE

DSTO

UNCLASSIFIED PUBLIC RELEASE

Mathematical Algorithms for Simulation

- Algorithms are iterative
  - Functions of the algorithm are executed in correct order once per iteration.



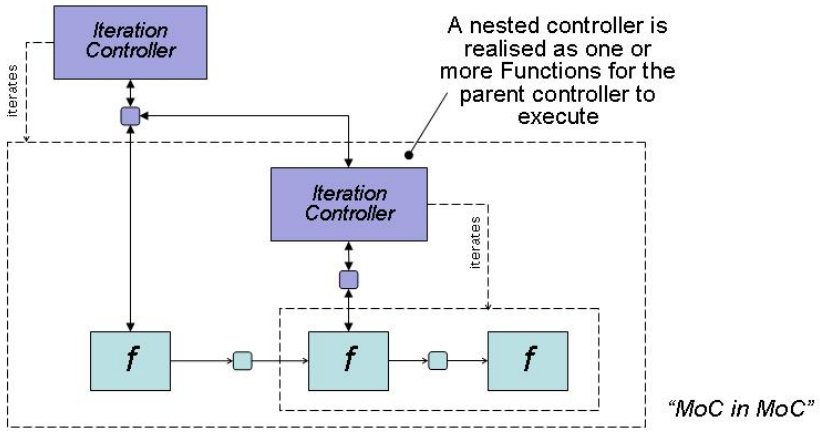
UNCLASSIFIED PUBLIC RELEASE

DSTO

UNCLASSIFIED PUBLIC RELEASE

Mathematical Algorithms for Simulation

- Algorithm iterations can be nested.

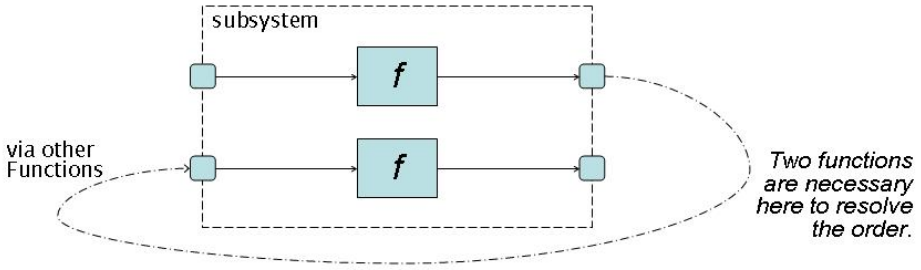


UNCLASSIFIED PUBLIC RELEASE

DSTO

Issue

- The logical conceptual decomposition of a system will not in general map one to one with a mathematical algorithmic decomposition of it.
  - Some subsystems will need as a minimum more than one Function.

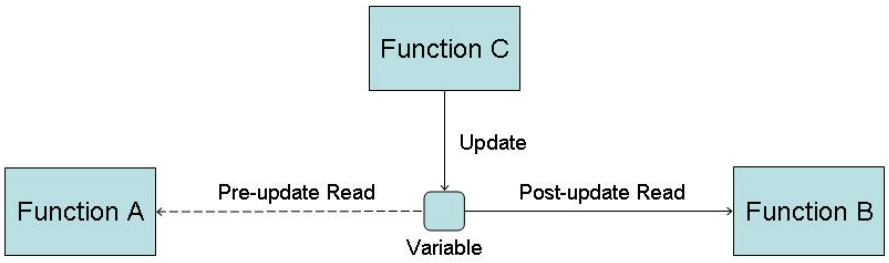


Some Definitions

- A Function is a arbitrary collection of mathematics that can only be executed once all its input Variables have been properly updated in the context of the current iteration.
- A Variable is an arbitrary complex data structure that, within the context of an iteration, is updated and read by Functions.
- A Model of Computation is a set of rules regarding the execution and management of user declared Functions and Variables, and is implemented by an Iteration Controller.

UNCLASSIFIED PUBLIC RELEASE

Iterative Algorithms



In the context of an iteration:

- Functions that read from a variable pre-update must be executed before it is updated.
- Functions that read from a variable post-update must be executed after it is updated.

UNCLASSIFIED PUBLIC RELEASE

DSTO

UNCLASSIFIED PUBLIC RELEASE

Australian Government
Department of Defence
Defence Science and Technology Organisation

Pulling it all Together

UNCLASSIFIED PUBLIC RELEASE

DSTO

Proposal

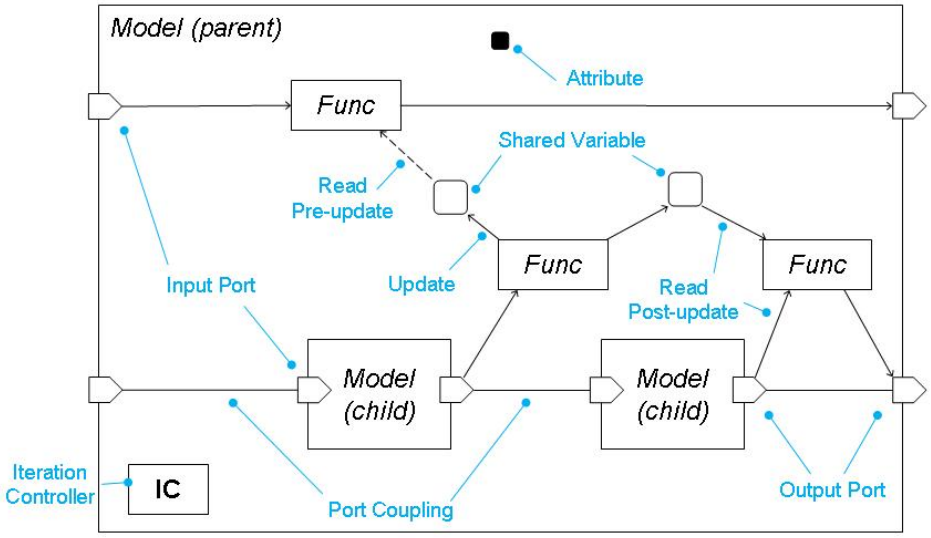
- Use a modified concept of a SysML Internal Block Diagram that support definitions of mathematical algorithms to represent continuous system behaviour (including generation of discrete events).

Model Symbology

Elements		Relationships	
	Model	→	Update
	Child Model Interface	→	Read Post-update
IC	Iteration Controller	⇢	Read Pre-update
	Function	—	Port Coupling
	Input Port (Variable)		
	Output Port (Variable)		
	Shared Variable		
	Attribute (Variable)		

UNCLASSIFIED PUBLIC RELEASE

Model Diagrams

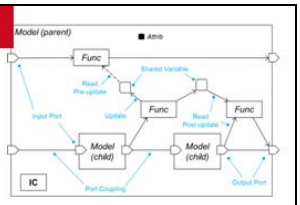


UNCLASSIFIED PUBLIC RELEASE

DSTO

UNCLASSIFIED PUBLIC RELEASE

Specification

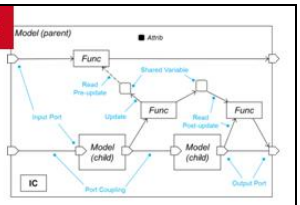


- Models are defined by:
  - An interface, which consists of any number of Input and/or Output Ports.
  - An internal definition consisting of any number of Functions, Shared Variables, Attributes, or interfaces of Child Models.
  - Relationships between Functions and Variables (including Ports of child Models).
  - Zero or One Iteration Controller.

UNCLASSIFIED PUBLIC RELEASE

DSTO

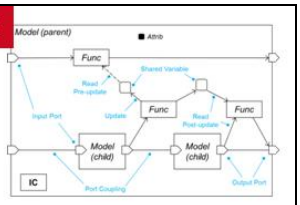
Specification



- Functions
  - Must have a means of indicating to Iteration Controllers which MoC they are dependent upon, if any.
- Variables
  - Attributes are constant Variables that are only updated when the owning model is instantiated.
    - All Functions have Read pre-update access to Attributes
  - Can support 'tags' that will have specific meaning to certain MoC.

UNCLASSIFIED PUBLIC RELEASE

Specification

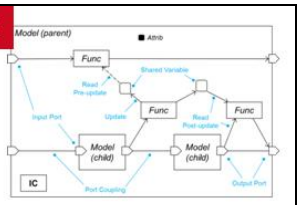


- Update Relationship
  - The source end must connect to a Function. The target end must connect to either a Shared Variable, parent Model's Output Port, or a child Model's Input Port.
- Read post-update Relationship
  - The target end must connect to a Function. The source end must connect to either a Shared Variable or Port.
- Read pre-update Relationship
  - The target end must connect to a Function. The source end must connect to either a Shared Variable or Port.

UNCLASSIFIED PUBLIC RELEASE

DSTO

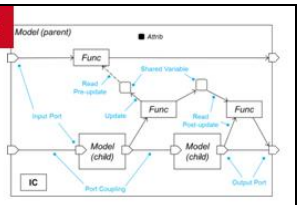
Specification



- Coupling Relationship
  - Connects either:
    - An Input Port of the parent Model to an Input Port of a child Model;
    - An Input Port of the parent Model to an Output Port of the parent Model;
    - An Output Port of a child Model to an Input Port of a child Model;
    - An Output Port of a child Model to an Output Port of the parent Model.
  - Copies updated content of the source Port (parent Model's input or child Model's output) to the target Port.

UNCLASSIFIED PUBLIC RELEASE

Specification



- Iteration Controller
  - Embodies a specific Model of Computation.
  - Coordinates the execution of a set of Functions within the Model in which it is declared according to relationships between Functions and Variables and specific MoC rules.
    - Selection of the set of Functions depends on the specific IC.
    - A Function dependent on a MoC cannot be executed under another MoC.
  - An IC must itself be expressed as one or more Functions that can be executed by a parent IC.

UNCLASSIFIED PUBLIC RELEASE
DSTO

UNCLASSIFIED PUBLIC RELEASE

Variable step Discrete Time MoC

- Definition: For time t in each iteration i conforming to the Discrete Time model of computation.

$$t_{i+1} - t_i > 0$$

- Functions may 'request' a time value for a future DT iteration.
  - The DT IC will determine the time of the next DT iteration as the minimum all requests.
- Discrete Event simulations can be formulated in this DT MoC.

UNCLASSIFIED PUBLIC RELEASE

DSTO

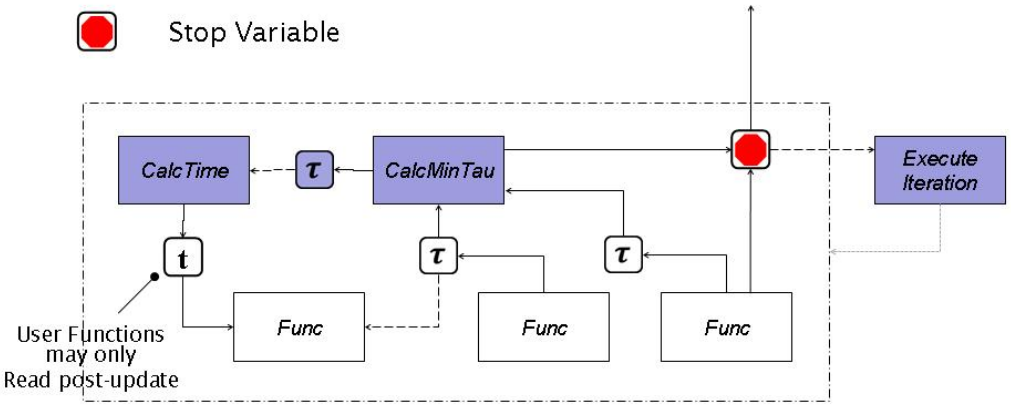
UNCLASSIFIED PUBLIC RELEASE

Elements supporting DT MoC

t Time Variable

τ Time Request Variable

Stop Variable



UNCLASSIFIED PUBLIC RELEASE

DSTO

UNCLASSIFIED PUBLIC RELEASE

ODE Solving MoC

- Definition: A coordinated solving of simultaneous ODEs according to the users choice of integration algorithm as applied to a selected integrators.
- Works within the DT MoC
  - Non-causal (eg. Runge-Kutta) algorithms result in multiple 'intermediate' iterations of a Functions that calculate derivatives. These iterations are ODE MoC specific.

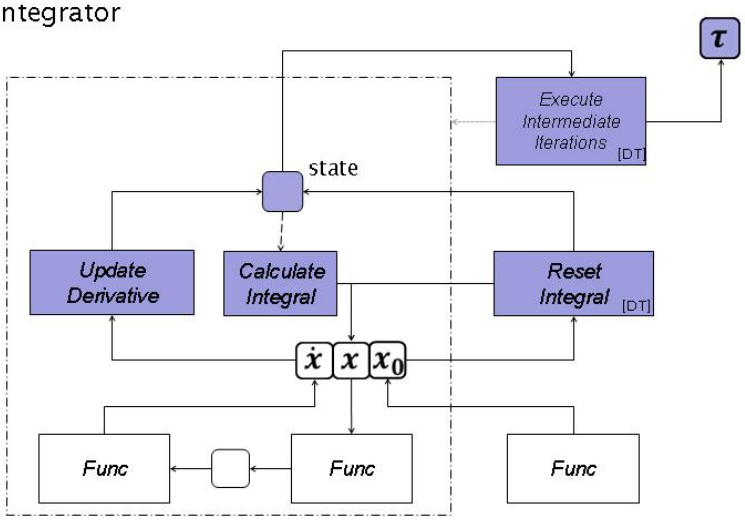
UNCLASSIFIED PUBLIC RELEASE

DSTO

UNCLASSIFIED PUBLIC RELEASE

Elements supporting ODE Solver MoC

$\dot{x}$ x x_0 Integrator

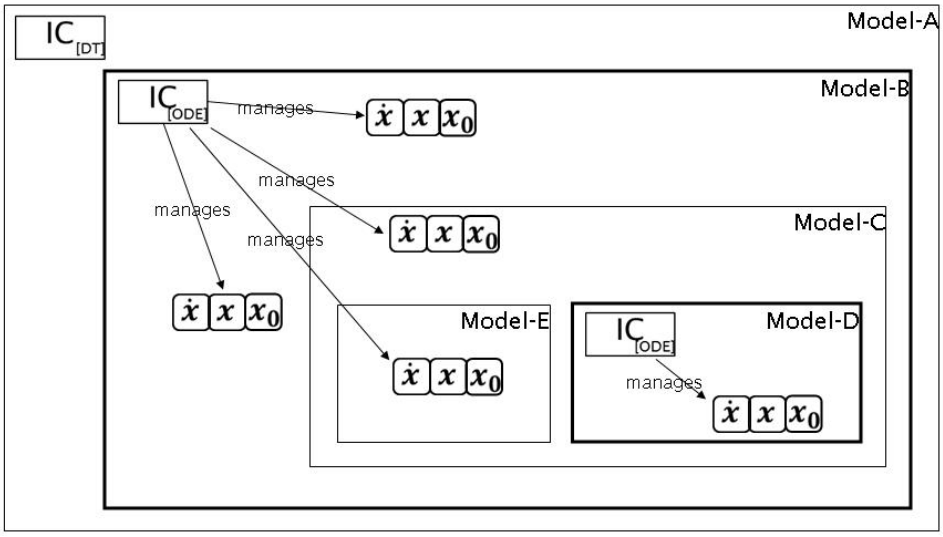


UNCLASSIFIED PUBLIC RELEASE

DSTO

UNCLASSIFIED PUBLIC RELEASE

Selection of ODE Sub-network of Functions



UNCLASSIFIED PUBLIC RELEASE

DSTO

UNCLASSIFIED PUBLIC RELEASE

Drawing the line between in-built MoC and User defined Model constructs

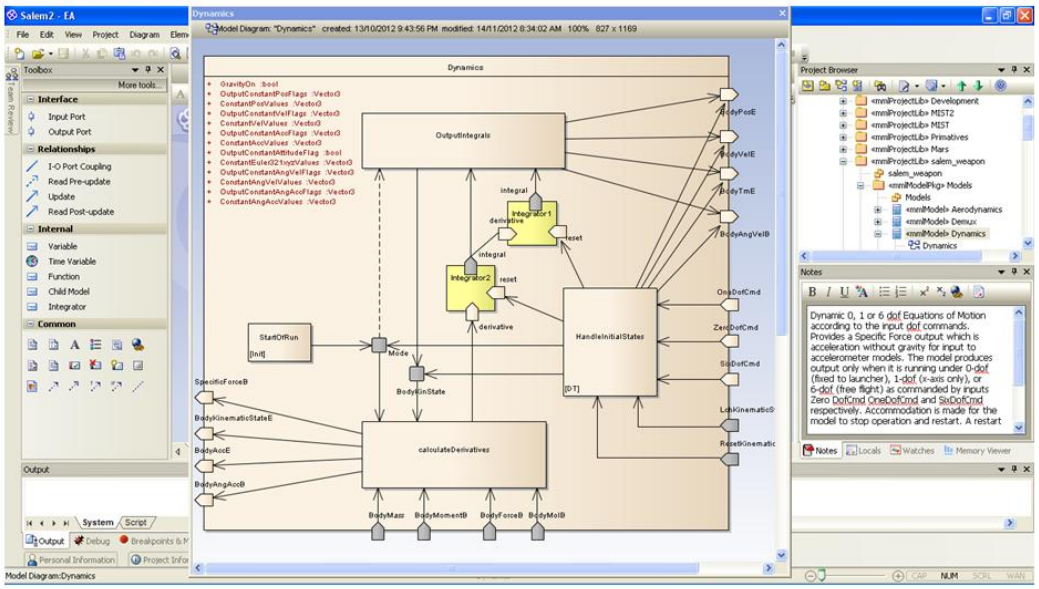
- There are many subtle algorithmic design patterns that if not built into an MoC must (if needed) be implemented by the modeller. Most are not particularly universal in application.
- Specific Function triggering (over and above MoC dependencies):
  - Input Variable update;
  - Time = Tau (in DT MoC).
- Function or Model enabling/disabling.
- Automatic clearing of data from variables between iterations.

UNCLASSIFIED PUBLIC RELEASE

DSTO

UNCLASSIFIED PUBLIC RELEASE

Walking the walk



UNCLASSIFIED PUBLIC RELEASE

DSTO

UNCLASSIFIED PUBLIC RELEASE

FIN

Questions

UNCLASSIFIED PUBLIC RELEASE

DSTO